Chapitre 2

AUTOMATES A ETATS FINIS

1. Introduction

Un reconnaisseur pour un langage est un programme qui prend en entrée une chaîne x et répond "oui" si x est une chaîne du langage et "non" autrement. On compile une expression régulière en un reconnaisseur en construisant un diagramme de transition généralisé appelé automate fini. Un automate fini peut être déterministe ou non déterministe, ce dernier terme signifiant qu'on peut trouver plus d'une transition sortant d'un état sur le même symbole d'entrée.

- Il y a conflit temps/ place :

- Les automates finis déterministes produisent des connaisseurs plus rapides que les automates finis non déterministes.
- Un automate fini déterministe peut être beaucoup plus volumineux qu'un automate fini non déterministe.

2. Automates à états finis non déterministes

Un automate fini non déterministe (AFN en abrégé) est un modèle mathématique qui consiste:

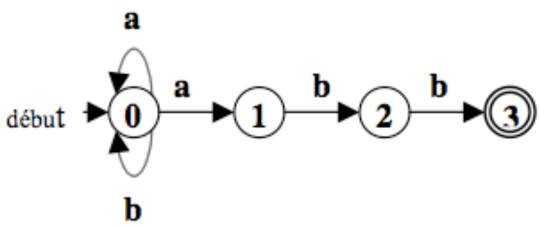
1. Un ensemble d'états E ;
2. Un ensemble de symboles d'entrées S (l'alphabet des symboles d'entrée) ;
3. Une fonction Transiter, qui fait correspondre des couples état-symbole à des ensembles d'états ;
4. Un état e0 qui est distingué comme état de départ ou état initial ;
5. Un ensemble d'états F distingués comme états d'acceptation ou états d'acceptation ou états finals.

Un AFN peut être représentée graphiquement comme un graphe orienté étiqueté, appelé graphe de transition, dans lequel les nœuds sont les états et les arcs étiquetés représentent la fonction de transition.

Remarque : Ce graphe ressemble à un diagramme de transition, mais le même caractère peut étiqueter deux transitions au plus en sortie d'un même nœud et les arcs peuvent être étiquetés par le symbole spécifique au même titre que les symboles d'entrées.

Exemple: Soit le langage dénoté par l'expression régulière (a|b)*abb, consistant en l'ensemble des chaînes de a et de b se terminant par abb.

La figure ci-dessous représente le graphe de transition par un AFN qui reconnaît ce langage. L'ensemble des états de l'AFN est {0, 1, 2, 3} et l'alphabet d'entrée est {a, b}. L'état 0 est distingué comme étant l'état de départ et l'état d'acceptation 3 est indiqué par un double cercle.



Automate fini non déterministe

Implantation d'un AFN

En machine la fonction de transition d'un AFN peut être implantée à l'aide d'une table de transition dans laquelle il y a une ligne pour chaque état et une colonne pour chaque symbole d'entrées et e, si nécessaire.

L'entrée pour la ligne i et le symbole a dans la table, donne l'ensemble des états qui peuvent être atteints par une transition depuis l'état i sur le symbole a (voir figure suivante).

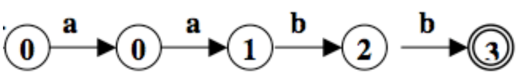
Etat	SYMBOLE D'ENTREE	
	A	b
0	{0, 1}	{0}
1	-	{2}
2	-	{3}
3	-	-

Table de transition du dernier AFN

Un AFN accepte une chaîne d'entrée x si et seulement si il existe un certain chemin dans le graphe de transition entre l'état de départ et un état d'acceptation, tel que les étiquettes d'arcs le long de ce chemin épellent le mot x.

L'AFN de la figure 2 accepte les chaînes d'entrées abb, aabb, babb, aaabb, … Par exemple, aabb est acceptée par le chemin depuis 0, en suivant de nouveau l'arc étiqueté a vers l'état 0, puis vers les états 1,2 et 3 via les arcs étiquetés a;b et b respectivement.

Un chemin peut être représenté par une suite de transitions d'état appelée déplacements. Ce diagramme montre les déplacement réalisées en acceptant la chaîne d'entrée aabb.



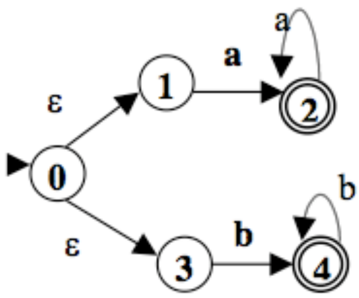
En général, il existe plus d'une suite de déplacements pouvant mener à l'état d'acceptation

Bien d'autres suites de déplacements pourraient être faites sur la chaîne d'entrée aabb, mais aucune des autres n'arrive à terminer dans un état d'acceptation.

Cette suite stationne dans l'état de non acceptation 0.

<u>Définition</u> : Le langage défini par un AFN est l'ensemble des chaînes d'entrées qu'il accepte.

Exemple:



Automate fini non déterministe acceptant aa*|bb*

La chaîne aaa est acceptée en se déplaçant via les états 0, 1, 2, 2, et 2. Les étiquettes de ces arcs sont : ε, a, a, et a dont la concaténation est aaa. Notons que ε disparaît dans la concaténation.

3. Automates finis déterministes

Un automate fini déterministe (AFD en abrégé) est un cas particulier d'automate fini non déterministe dans lequel :

1. Aucun état n'a de ε transitions, c'est à dire de transition sur l'entrée ε et
2. Pour chaque état e et chaque symbole d'entrée a, il y a au plus un arc étiqueté a qui quitte e.

Un automate fini déterministe a au plus une transition à partir de chaque état sur n'importe quel symbole.
Chaque entrée de sa table de transition est un état unique.

⇒ Il est très facile de déterminer si un automate fini déterministe accepte une chaîne d'entrée, puisqu'il existe au plus un chemin depuis l'état de départ étiqueté par cette chaîne.

- L'algorithme suivant montre comment simuler le comportement d'un AFD sur une chaîne d'entrée.

Algorithme: Simulation d'un AFD

Données : Une chaîne d'entrée x terminée par un caractère de fin de fichier fdf.
Un AFD D avec un état de départ e_0 et un ensemble d'états d'acceptation F.

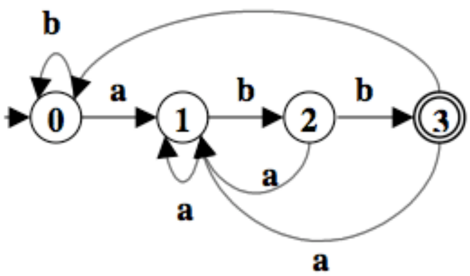
Résultat : La réponse "oui" si D accepte x; "non" dans le cas contraire.

Méthode : La fonction transiter(e, c) donne l'état vers lequel il y a une transition depuis l'état e sur le caractère d'entrée c. La fonction CarSuiv retourne le prochain caractère de la chaîne d'entrée x.

```
e:=e0
C:=CarSuiv();
Tanque c ≠ fdf faire
            e:= Transiter(e, c);
            c:=  CarSuiv();
fin
si e ∈ F alors
      retourner "oui"
sinon
      retourner "non"
```

Exemple : La figure suivante présente le graphe de transition d'un automate fini déterministe qui accepte la langage (a|b)* abb.

-Avec cet AFD et la chaîne d'entrée ababb, l'algorithme passe par la suite les états 0, 1, 2, 1, 2 et 3 et retourne oui.



AFD acceptant (a|b)*abb

4. Conversion d'un AFN en un AFD

Si on utilise un AFN pour vérifier si une chaîne x ∈ langage, on trouvera plusieurs chemins qui épellent la même chaîne x, on doit les avoir tous pris en compte avant d'en trouver un qui mène à l'acceptation ou de découvrir qu'aucun d'entre eux ne mène à un état d'acceptation.

<u>**Remarque**</u> : Dans la table de transition d'un AFN, chaque entrée est un ensemble d'états. Dans la table de transition d'un AFD chaque entrée est un état unique.
L'idée ⇒ chaque état de l'AFD correspond à un ensemble d'états de l'AFN.

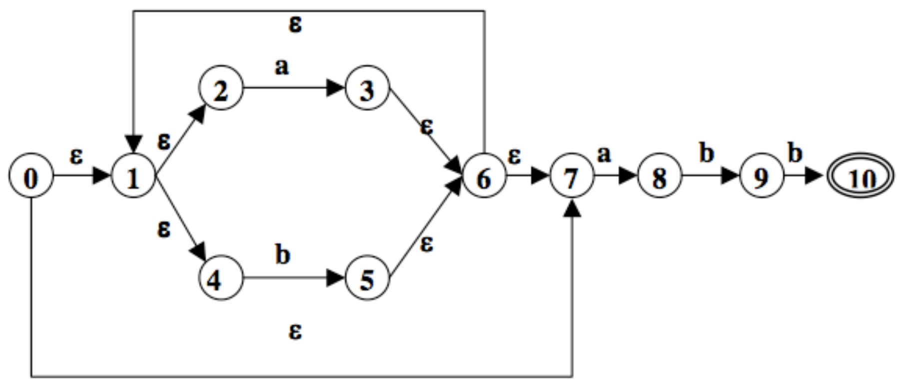
<u>Algorithme</u> : Construction d'un AFD à partir d'un AFN

Données : Un AFN N
Résultat : Un AFD D qui accepte le même langage.
Méthode : Notre algorithme construit une table de transition Dtran pour D.
Chaque état de l'AFD est un ensemble d'états de l'AFN et on construit Dtran de telle manière que D simulera "en parallèle" tous les déplacements possibles que N peut effectuer sur une chaîne d'entrée donnée.

On utilise les opérations suivantes pour garder trace des ensembles d'états de l'AFN.
E représente un état de l'AFN et T représente un ensemble d'états de l'AFN.

Opération	**Description**
ε -fermeture(e)	Ensemble des états de l'AFN accessibles depuis un état e de l'AFN par de ε -transitions uniquement
ε -fermeture(T)	Ensemble des états de l'AFN accessibles de puis un état e appartenant à T par des ε -transitions uniquement
Transiter(T, a)	Ensemble des états de l'AFN vers lesquels il existe une transition sur le symbole à partir d'un état e appartenant à T

Opérations sur les états d'un AFN



Un NFA pour le langage (a|b)*abb

ε-fermeture (0) = {0, 1, 2, 0, 4, 7} , Transiter ({0, 1, 2, 4, 7}, a) = { 3, 8}
ε-fermeture({3, 8}) = {1, 2, 3, 4, 6, 7, 8}

Avant de voir le premier symbole d'entrée, N peut appartenir à n'importe lesquels des états de l'ensemble ε-fermeture(e_0) ou e_0 est l'état de départ de N (il peut être dans T = {0, 1, 2, 4, 7}). Supposons que a est le prochain symbole d'entrée. Quand il voit a, N peut passer dans l'un des états de l'ensemble Transiter (T, a)= Transiter({0, 1, 2, 4, 7}, a)={3, 8}.

Quand on autorise des ε-transitions, N peut être dans un des états de ε-fermeture(Transiter(T, a)), après avoir lu le a= ε-fermeture({3, 8})={1, 2, 3, 4, 6,7, 8}.

Remarque (Ceci est le même si on a déjà lu une partie de la chaîne et on est arrivé a un caractère a à traiter).

On construit Détats, l'ensemble des états de D et Dtran, la table de transition de D, de la manière suivante :

-Chaque état de D correspond à un ensemble d'états de l'AFN dans lesquels N pourrait se trouver après avoir lu une suite de symboles d'entrée, en incluant toutes les ε-transitions possibles avant ou après la lecture des symboles.

L'état de départ D est ε-fermeture (e_0). On ajoute des états et des transitions à D' en utilisant l'algorithme 3.

Algorithme: Construction des sous-ensembles

```
Au départ ε-fermeture(e0) est l'unique état de Détats et il est non marqué;
Tant que il existe un état non marqué T dans Détats faire début
        Marquer T;
                Pour chaque symbole d'entrée a faire début
          U : = ε-fermeture(Transiter(T, a));
          Si U n'appartient pas à Détats alors
          Ajouter U comme nœud non marqué de Détats;
          Dtran[T,a] : = U
                Fin
Fin
```

Un état D est un état d'acceptation si c'est un ensemble d'états de l'AFN qui contient au moins un état d'acceptation

Exemple :

Appliquons l'algorithme

L'état de départ de l'AFD équivalent est ε-fermeture(0), qui est A = {0,1,2,4,7}, étant donné que ce sont précisément les états accessibles depuis l'état 0 via un chemin dans lequel chaque arc est étiqueté ε.

- L'alphabet des symboles d'entrée est ici {a,b}. L'algorithme précédent nous dit de marquer A et de calculer e-fermeture(Transiter(A, a)). Nous calculons d'abord Transiter (A,a), l'ensemble des états de N qui ont des transitions sur a depuis les éléments de A. Parmi les états 0,1,2,4 et 7, seuls 2 et 7 ont de telles transitions vers 3 et 8.

Aussi :

ε-fermeture(Transiter({0,1,2,4,7},a)=ε-fermeture({3,8})={1,2,3,4,6,7,8}

Appelons cet ensemble B. Nous avons alors Dtran[A,a]= B.

Parmi les états de A, seul 4 a une transition sur b vers 5 aussi l'AFD a une transition sur b depuis A vers :

C = ε-fermeture({5}) = {1,2,4,5,6,7}. Donc Dtran[A,b]=C

Etat	Symbole d'entrée	
	A	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

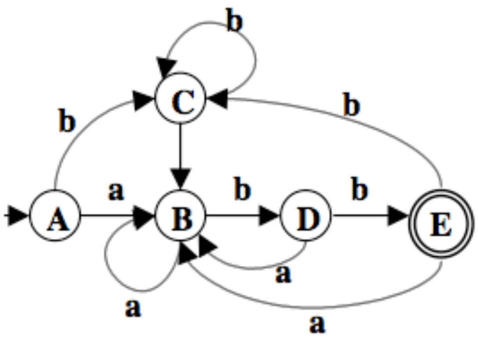
Table de transition Dtran pour le DFA

- Si nous continuons ce processus avec les ensembles actuellement non marqués B et C, on atteint finalement le point où tous les ensembles qui sont des états de l'AFD sont marqués.

- Les cinq différents ensembles que nous construisons réellement sont
  A={ 0,1,2,4,7} D={1,2,4,5,6,7,9}
  B={1,2,3,4,6,7,8} E={1,2,4,5,6,7,10}
  C={1,2,4,5,6,7}

L'état A est l'état de départ et l'état E est l'unique état d'acceptation.

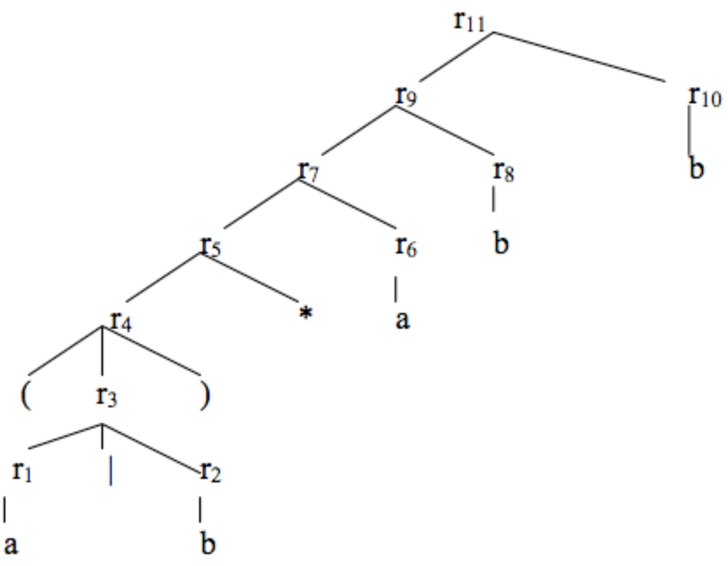
Voici le graphe de transition correspondant à l'AFD



AFD pour (a|b)*abb

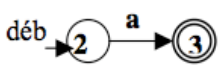
5. Conversion d’une ER en un AFN

La conversion est basée sur la construction de thompson illustrée ci-dessous sur un exemple.
Soit l'expression régulière r = (a|b)* abb.
La figure suivante présente un arbre de décomposition de r.

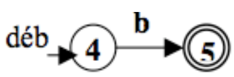


Décomposition de (a|b)*abb

pour le composant r_1, le premier a, on construit l'AFN

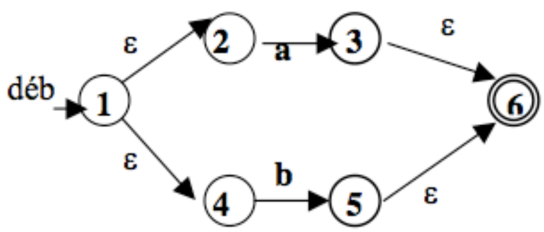


pour r_2, on construit

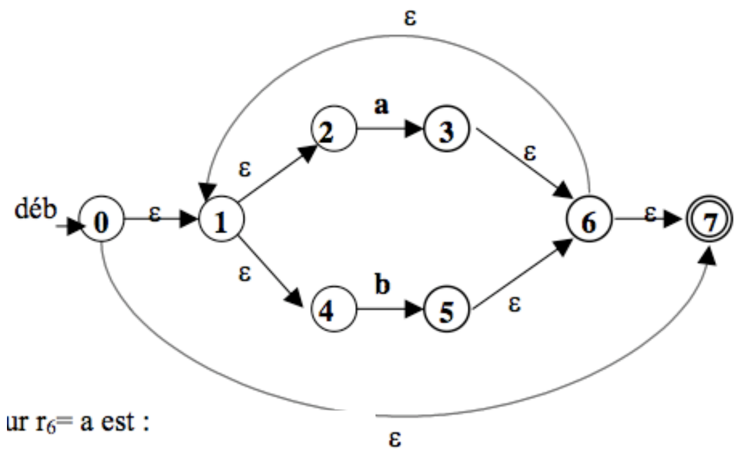


On peut maintenant combiner $N(r_1)$ et $N(r_2)$ en utilisant la règle d'union pour obtenir l'AFN. $N(r_1)$ est l’AFN de r_1.

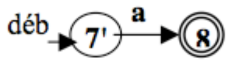
pour $r_3=r_1|r_2$:



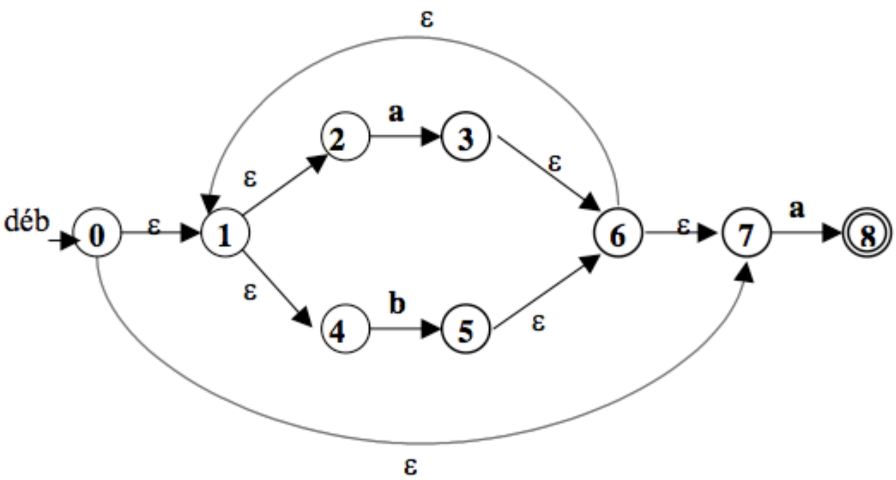
L'AFN pour r_4 est le même que celui pour r_3. L'AFN pour $(r_4)^*$ est alors :



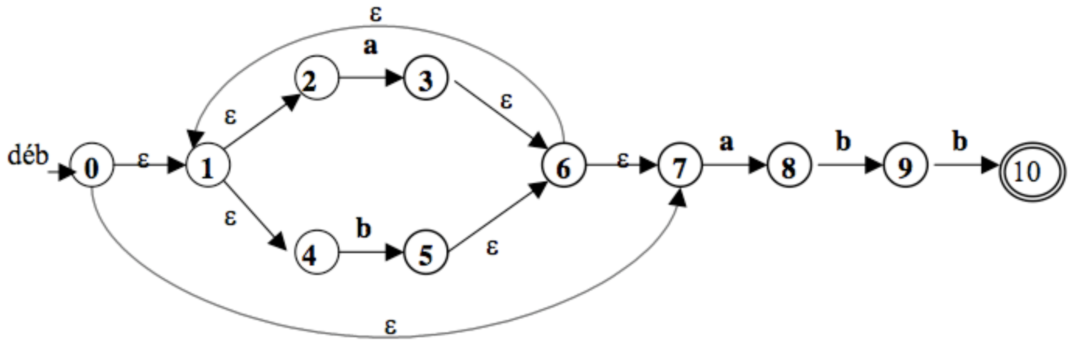
L'AFN pour r_6 est :



Pour obtenir l'automate pour $r_7 = r_5r_6$, on fusionne les états 7 et 7' en appelant l'état résultant 7 pour obtenir



En continuant ainsi on obtient l'AFN pour $r_{11}=(a|b)^*abb$



AFN obtenu par la construction de Thompson pour r11=(a|b)*abb